INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Peer-to-peer Semantic Wikis

Hala Skaf-Molli — Charbel Rahhal — Pascal Molli

N° 6714

October 08

Thème COG

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--6714--FR+ENG

Peer-to-peer Semantic Wikis

Hala Skaf-Molli [*] , Charbel Rahhal[†], Pascal Molli [‡]

Thème COG — Systèmes cognitifs
Projet ECOO

Rapport de recherche n° 6714 — October 08 — 31 pages

Abstract: This paper presents SWOOKI the first peer-to-peer semantic wiki. A P2P semantic wiki combines advantages of both semantic wikis and P2P Wikis. A P2P semantic wiki network is composed of a set of interconnected autonomous servers that can dynamically join and leave the network. It is a truly decentralized unstructured P2P system which does not require any central coordination or knowledge. It relies on a symmetric communication model where every peer may act as both a server and a client. Every peer hosts a replica of shared data. The total replication improves data availability and performance. It allows to query and access data locally without any data transfer. Furthermore, it enables off-line works and transactional changes. A replicated groupware system is correct if it maintains the convergence, causality preservation and intention preservation. The CCI correction model is widely defined for linear structures *i.e.* text. None of the exiting synchronization algorithms takes in consideration a mix of text and semantic data as the case of semantic wikis. This paper defines a data model for P2P semantic wikis and its associated editing operations. It defines also the intentions of the operations of this new data type. The paper extends the WOOTO algorithm to synchronize and ensure the CCI model for this new data type. The algorithm is implemented in a fully operational system.

Key-words: P2P, Semantic Wikis, Optimistic replication, Collaborative writing

[*] Skaf@loria.fr,, ECOO Project, Nancy-University, LORIA, INRIA Centre - Nancy Grand Est
[†] charbel.rahal@loria.fr, ECOO Project, Nancy-University, LORIA, INRIA Centre - Nancy Grand Est
[‡] molli@loria.fr, ECOO Project, Nancy-University, LORIA, INRIA Centre - Nancy Grand Est

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy (France)
Téléphone : +33 3 83 59 30 00 — Télécopie : +33 3 83 27 83 19

Wikis Sémantiques Pair-à-pair

Résumé : Dans ce papier, nous présentons SWOOKI le premier wiki sémantique pair-à-pair. Un wiki sématique pair-à-pair combine les avantages des wikis sémantiques et ceux des wikis pair-à-pair. Le réseau du wiki sémantique pair-à-pair est composé d'un ensemble de serveurs autonomes (pairs) interconnectés. Ces pairs peuvent joindre et quitter le réseau à tout moment. Ce réseau pair-à-pair est non structuré et ne nécessite aucune coordination centrale. Il est basé sur un modèle de communication symértique où chaque pair joue les deux rôles client et serveur. Chaque pair possède une réplique des données partagées. La réplication totale de données améliore leur présence et la performance du système. Elle permet d'exécuter des requêtes et d'accéder aux données localement sur n'importe quel pair sans générer du transfert de données entre les pairs. La réplication totale permet aussi un mode d'édition déconnecté et des changements transactionnelles des pages wiki. Un système groupware replicant des données est dit correct s'il respecte le modèle CCI c.à.d s'il maintient la convergence, s'il preserve la causalité et l'intention. Le modèle CCI a été défini pour des structures linéaires i.e. texte. Le contenu des pages dans les wikis sémantiques combine du texte et des annotations sémantiques. Acutellement, il n'existe aucun algorithme sur pair-à-pair pour la synchronisation de ce nouveau type de données. Ce papier définit un modèle de données pour les wikis sémantiques pair-à-pair et les opérations d'édition associées. Il définit aussi les intentions des opérations sur ce nouveau type de données. Ce papier extensie l'algorithme WOOTO afin de synchroniser et d'assurer le respect du modèle CCI pour ce type de données. Une implémentation de cet algorithme a été réalisée.

Mots-clés : Pair-à-pair, Wikis sémantiques, Réplication optimistique, Édition collaborative

Contents

1 Introduction

Wikis are the most popular tools of Web 2.0, they provide easy way to share and contribute to global knowledge, the encyclopedia Wikipedia is the famous example of a wiki system. In spite of their fast success, wiki systems have some drawbacks. They suffer from search and navigation [34], it is not easy to find information in wikis [6]. They have also scalability, availability and performance problems [35, 18] and they do not support offline works and atomic changes. To overcome these limitations, wiki systems have evolved in two different ways: semantic wikis and peer-to-peer wikis.

Semantic Wikis Semantic wikis integrate the Semantic Web technologies such as RDF [2] and SPARQL[3] to improve the wikis structures, the search and the navigation between pages. This makes the wiki content machine-readable allowing content extraction for reuse in external applications. Many semantic wikis are being developed[34, Semantic MediaWiki], [25, IkeWiki] and [6, SweetWiki]. There are two approaches of semantic wikis [6]:

- *The use of ontologies for wikis*: requires the load of an existing ontology. The advantage is to build controlled vocabularies but it can be too rigid for emergent domains where ontologies are not clearly defined.
- *The use of wikis for ontologies*: semantic wikis let users choose their own vocabularies [34]. Semantic annotations are integrated directly in the wiki text. Semantic data appear in their context. The main advantage is to allow the emergence of an ontology.

P2P wikis Peer-to-peer wikis is another interesting evolution of wiki systems. They provide faults-tolerance, better scalability, infrastructure cost sharing, and deliver better performance than centralized wikis by replicating wiki pages on different wiki servers. The replication is "total" if all pages are replicated on all servers [35, Wooki], [15, Repliwiki], [1, git-wiki]. It is "partial" if a single page is replicated few times [18, DistrWiki], [11, DtWiki] and Piki [21, Piki]. Partial replication is generally implemented on Distributed Hash tables (DHT), total replication is generally implemented on unstructured P2P networks. Total replication is costly but it enables offline editing, transactional changes and local requests execution. Users can work disconnected if they have no internet connection or if they decide to disconnect directly from the user interface. While disconnected, a user can change many semantic wiki pages in order to produce a consistent change. By this way she generates some sort of transactional changes. Total replication is already used in other P2P collaborative systems such as Usenet[27] or distributed version control systems.

The main advantages of the partial replication approach are its support to a virtual infinite storage, it generates less traffic than the total replication approach and provides an easy join of a new site to the P2P network. In partial replication, adding a peer in the network increases storage capacity. This is not the case in total replication as

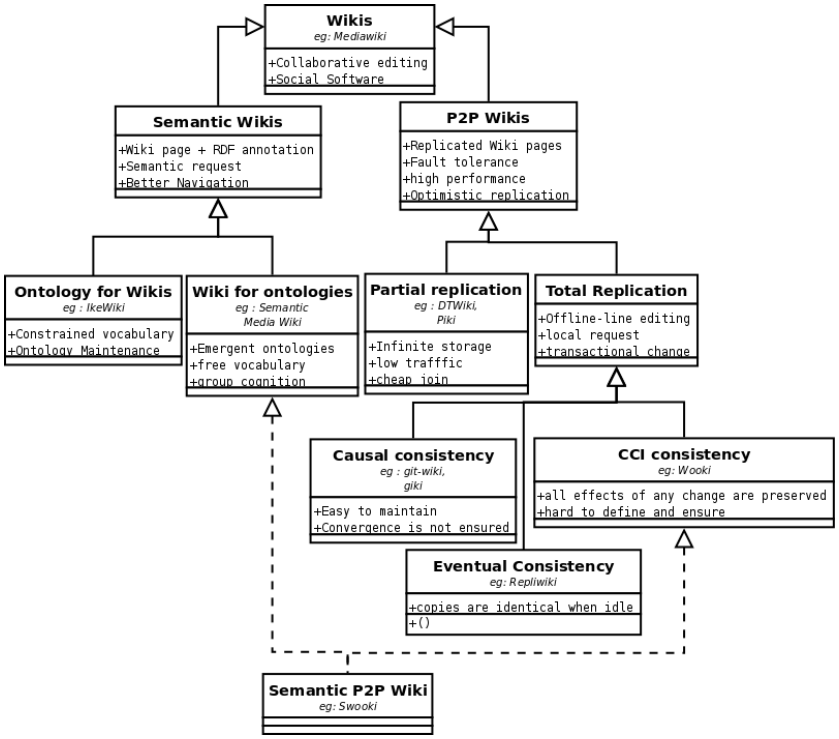

Figure 1: Wiki, Semantic Wiki and P2P Wiki

all data are replicated in all nodes. In partial replication, when a change is made on a copy, it has to be propagated only to few copies. In total replication, a change has to be propagated to all sites. Consequently, partial replication generates less traffic than total replication. Finally, when a new peer is joining the network, total replication proceeds to a state transfer with an active peer. The state transfer can be important in size. In partial replication, just few pages need to be replicated within the P2P network. However, pages requests are more complex in partial replication, offline editing and transactional changes are much more difficult to achieve.

In all cases (partial or total replication), replication requires a consistency model to maintain replicas. Optimistic replication approach is more suitable for P2P systems than pessimistic replication [24]. Optimistic replication is compatible with P2P constraints: it *scales, delivers better performances and better disconnections*. It tolerates the replicas diverge and synchronizes copies from time to time. In optimistic replication, different levels of consistency are defined. Some systems adopt causal consistency [1, git-wiki], others ensure eventual consistency [15, repliwiki]. Most advanced systems ensure the CCI model that stands for Causality, Convergence and Intention

preservation [35, Wooki]. This last consistency model has been developed by the CSCW community for building replicated group editors. *Causal consistency* ensures that all sites have seen the same causal history of changes but does not ensure that all copies converge. *Eventual consistency* ensures that all copies are identical when the system is idle. CCI model ensures causal consistency, eventual consistency and intention preservation. *Intention preservation* means that in the convergence state all changes are present and achieve the same effects observed by the user when she made her original changes.

In this paper, we focus on the building of a peer-to-peer semantic wiki. Such a system combines advantages of P2P wikis and Semantic wikis. Our system is a combination of **the use of wikis for ontologies** approach such as Semantic MediaWiki and a **peer-to-peer wiki** based on **total replication** and **CCI consistency** such as Wooki (see figure 1). All other combinations are possible, however, this combination has several advantages:

- *The use of wikis for ontologies*: semantic wikis generate group cognition [30]. Wikis can be use for collaborative knowledge building.
- *The use of wikis for ontologies*: semantic wikis embed semantic annotations in wiki text. The position of the semantic annotation within the wiki text is important. It allows humans to better understand the origin of the annotation and maybe to modify it.
- Total replication in P2P wiki systems allows to query, access and retrieve data locally from any peer without the need for search mechanisms nor transfer of the semantic data between peers to resolve queries [7, 19]. Therefore, "querying the network" [29] becomes querying any peer.
- Total replication enables transactional changes *i.e.* atomic changes across multiple pages. Supporting transactional changes is a very important feature in the context of semantic wiki. In semantic wiki, a wiki page presents a concept of an ontology, so a modification of one concept may require atomic changes to other concepts. This feature is critical especially if we want to use semantic wiki for collaborative ontologies development. We can make the parallel with softwares development environment where it is necessary to allow atomic change to a group of files.
- Transactional changes can be considered as a new editing mode. It is often called "insulated work" or "multi-synchronous editing". This mode is very common in software engineering for writing programs. If integrated in semantic wiki systems, it can be used for ontology engineering.

Building a semantic P2P wiki based on the CCI model is challenging. **The fundamental problem is to provide an optimistic replication algorithm that (1) is compatible with P2P constraints, (2) ensures the CCI model and (3) manages a semantic wiki page.**

Many algorithms have been proposed in the optimistic replication domain. Just few are compatible with P2P constraints. The churn of the P2P network where sites join and leave continuously is a critical requirement for replication. Replication frameworks such as [16, IceCube] or [26, ACF] fail to satisfy this requirement.

The second requirement is the CCI consistency model. The CCI consistency model is defined as follow:

1. *Convergence:* peers with the same set of editing operations compute the same states for the replicated data.

2. *Causality preservation:* operations ordered by a precedence relation will be executed in the same order on every peer.

3. *Intention and Intention preservation:* the intention of an operation is the effects observed on the state when the operation was generated, *Intention preservation* as defined by Sun et al. [31] is "for any operation *op*, the effects of executing *op* at all sites are the same as the intention of *op* and the effect of executing *op* does not change the effects of independent (not causally dependent) operations".

P2P optimistic replication algorithms often ensure only eventual consistency. They do not ensure that intentions are preserved. For example, in [14] or in [22, Bayou], it is possible to converge to a state where some changes have been ignored. In this case, the system itself can generate lost updates. This is unacceptable for collaborative editing. Finally, for the replication algorithms that ensure CCI and are compatible with P2P constraints, they have to support a new data type : a semantic wiki page. MOT2 [8] can potentially do that. However, MOT2 relies on transformation functions to really merge concurrent updates. Transformation functions are challenging to write and there are no defined ones for semantic wiki data type. WOOT[23] is also another algorithm compatible with P2P constraints and ensures the CCI model. However, WOOT is not generic and just handle linear structures. Semantic graphs cannot be managed by WOOT.

In this paper, we define formally the semantic wiki page data type, we specify its operations and we define the intentions of these operations. We extend the WOOT algorithm in order to take into account semantic annotations and finally we build the first P2P semantic wiki based on this algorithm called SWOOKI.

The paper is organized as follows. The section 2 presents some related works. The section 3 details the general approach for building a P2P semantic wiki. It defines a data model and editing operations. The section 4 defines the causality and intentions of operations used to edit semantic data. The section 5 develops the integration algorithm. The section 6 gives an overview of the architecture. The last section concludes the paper and points to future works.

2 Related work

As pointed out in the introduction, **the fundamental problem for building a P2P semantic wiki is to provide an optimistic replication algorithm that (1) is compatible with P2P constraints, (2) ensures the CCI model and (3) manages a semantic wiki page as defined in Semantic Media Wiki.**

Since a semantic wiki page embeds semantic annotations in the wiki text, these annotations can be extracted from the wiki pages. Different algorithms are needed to synchronize text and annotation. For instance, RDFSync [17,] for RDF data and algorithms such as Three-way-merge in Version Control Systems [4] can be used to synchronize text.

We examine existing solutions according to our criteria : collaboration, P2P constraints, CCI consistency and semantic wiki page data type.

2.1 Replication RDF resources in the P2P semantic web

Many researches have been done in P2P semantic web to share, query and synchronize RDF resources [19, 17, 7, 33, 9, 29].

RDFGrowth [33] and Publish/Subscribe Networks [9] focus on semantic data sharing where only one peer can modify the shared knowledge while others can read them. However, sharing is different from collaboration. In sharing, some peers publish data while others can only read these data and concurrent updates are not managed. In collaborative writing, some peers publish data, others can read and write these data and a synchronization algorithm integrates concurrent updates. Collaborative writing improves the quality of data, the experience of the collaborative Wikipedia demonstrates this. Moreover, interactions during collaborative writing enables group cognition [30] which is not possible with sharing interactions.

Edutella [19] proposes a RDF-based metadata infrastructure for P2P applications. It focuses on querying RDF metadata stored in distributed RDF repositories. The authors of Edutella proposes a replication service which "complements local storage by replicating in additional peers to achieve metadata persistence / availability and workload balancing while maintaining metadata integrity and consistency." However, they do not mention how to replicate and synchronize metadata. In this paper, we propose a framework to replicate and synchronize text and metadata while maintaining consistency of these data.

RDFSync [17] and RDFPeers [7] propose to synchronize RDF data. RDFSync [17] synchronizes a target RDF graph with a source one. RDF graphs are decomposed unequivocally into minimal subsets of triples (Minimum Self-Contained Graphs MSGs) and canonically represented by ordered lists of the identifies (hashes) of its composing MSGs. The synchronization algorithm performs a diff between the source and the target of the ordered list of MSGs. RDFSync can perform different kinds of synchronization, in the Target Growth Sync (TGS) the target becomes equal to the merge of both graphs, in the Target Erase Sync (TES) the target deletes unknown information by the *source* and finally in Target Change Sync (TCS) the target becomes equal to the source.

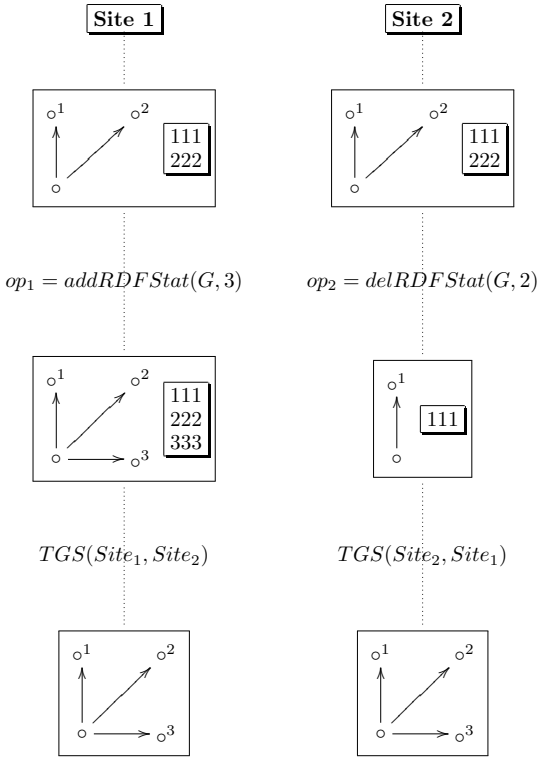

Figure 2: Concurrent editing with RDFSync

RDFSync can synchronize RDF data but it is not designed to synchronize linear data such as text, therefore, it can not be used alone to synchronize P2P semantic wikis. We can combine *rsync* and *RDFSync*, *rsync* for text and *RDFSync* synchronizes RDF data. This does not work because these algorithms do not respect the correction criteria (Causality, Convergence and Intention preservation) required by collaborative editing systems. For instance, consider two users on different peers modifying concurrently an initial RDF graph G which is decomposed into two MSGs with an ordered list of two hash numbers as shown in the figure 2. At $site_1$, a user adds a third statement, at the same time another user on $site_2$ deletes the second statement (see figure 2). With TGS strategy, $site_1$ needs nothing from the site $site_2$ meanwhile $site_2$ requests the second and the third statement from the first site. After the synchronization the convergence is ensured, both sites have the same RDF Graph, however, the intention of the user on the $site_2$ is not preserved since the second statement appears in the final result. This is because in *RDFSync* a deletion of a statement is effective only if all the peers delete that statement at the same time, this is not acceptable

in collaborative editing. In the same way, other synchronization strategies produce results that are not accepted during collaborative work.

RDFPeers [7] is a scalable distributed RDF repository. It is based on a structured P2P network. To enable faults-tolerance, RDFPeers uses partial replication of RDF data. Every RDF triple is stored at three nodes of the network. RDFPeers proposes a distributed query language to search RDF data. The approach of RDFPeers provides an infinite storage capacity for RDF data, however, it needs distributed querying mechanism. RDFPeers can not applied in the context of P2P semantic wikis because it does not handle text nor concurrent updates on replicas.

In summary, P2P Semantic Web researches focus on knowledge sharing and querying. In these systems, only one peer can modify the shared knowledge while others can read only, synchronization of concurrent updates is not taken in consideration. Supporting mutual modifications of replicated semantic data improves the quality of data and provides *reliable and scalable* semantic data repositories for P2P search.

2.2 Replication in Collaborative P2P systems

Data replication in collaborative P2P systems mainly relies on optimistic replication. A system is composed of n sites, each site replicates a copy of shared data. In order to modify a data, a site generates an operation on it. The operation is processed as follow:

1. It is immediately applied on the local site,
2. broadcasted to the other sites. We make the hypothesis that all operations eventually arrive.
3. The operation is received by others sites, maybe transformed in order to take into account concurrent operations and re-executed.

This model is used by many systems such as all version control systems, replicated group editors, lazy-replication in database systems etc.

Some existing P2P wikis such as *giki* or *git-wiki* use distributed version control systems (DVCS) to manage data. Wiki pages are stored as text files. DVCS manage them as code files. The main problem with DVCS is that there is no clear definition of their correctness. The only real property ensured by a DVCS is that each site maintains the same causal history of operations. By this way, it ensures causal consistency. Causal consistency implies that concurrent write operations can be seen in a different order on different machines. In this case, if two sites observe 2 write operations in different order then copies on both sites can diverge. DVCS systems are aware of this problem and delegate the problem to external merge algorithms for managing concurrent operations. However, as existing merge algorithms are not intrinsically deterministic, commutative and associative so convergence cannot be ensured in all cases.

Other systems such as *Usenet* [27] apply the Thomas write rule [14] to ensure eventual consistency. They ensure that, when the systems are idle *i.e* all operations have been sent

and received by all sites, all copies are identical. Unfortunately, if there is two concurrent write operations on the same data unit, then the rule of "the last writer wins" is applied and in this case, this means that a modification of a user is lost. Collaborative editing cannot be achieved if the system can loose some changes just to ensure eventual consistency.

The *Bayou* system [32] was proposed to support collaboration among users who cannot be or decide not to be continuously connected. Operations are broadcasted between sites using an epidemic propagation protocol. This is suitable for deploying a collaborative application on a peer-to-peer network. Unfortunately, in order to ensure convergence of copies, *Bayou* has to arrange eventually operations in the same order. To achieve this, it relies on a primary site that will enforce a global continuous order on a growing prefix of history. Using such a primary site may constitute a congestion point, and, anyway it is not suitable in a peer-to-peer system. In the Operational Transformation (OT) approach [31], such situation is qualified as an intention violation. It means that a user has observed an effect when making an update, and this effect cannot be observed on the convergence state. This is the reason that lead the OT community to develop the CCI model.

Many algorithms have been developed by the OT community such as SOCT2, GOTO, COT etc. They are designed to verify the CCI model. But only few of them support P2P constraints such as MOT2. In all cases, such algorithm requires to write transformation functions in order to handle concurrent operations. These transformation functions must verify some properties in order to ensure convergence and must be written in order to ensure intentions. Potentially, it is possible to write transformation functions for a semantic wiki page and use the MOT2 algorithm to integrate them. However, it is not done and the MOT2 algorithm suffers from a high communication complexity [13].

Wooki [35] is a P2P wiki system based on total data replication. Wiki pages are replicated over all the nodes. The pages copies are maintained by an optimistic replication mechanism that disseminates changes and ensures consistency. The originality of Wooki is its integration algorithm WOOTO (WOOTO is an optimized version of WOOT [20]). WOOTO is a peer to peer synchronization algorithm for linear structures such as strings, text and sequences. In Wooki, a wiki page is considered as a sequence of lines, WOOT ensures causality preservation, convergence and intention preservation properties for wiki pages.

To ensure convergence and preserve intention, WOOTO linearizes the dependency graph among the successively inserted and deleted lines and guarantees that the linearisation order is the same on all servers whatever is the delivery order of operations [20, 35].

Today, WOOTO is the only available P2P synchronization algorithm that ensures the CCI properties for linear data and does not produce a lot of messages traffic [13]. However, WOOTO can not be applied directly to a P2P semantic wiki. WOOTO is designed to synchronize linear structures, it can not synchronize a mix of text and RDF graphs. We propose to extend WOOTO to handle collaborative writing on replicated RDF data model.

3 P2P Semantic Wiki Approach

This section represents the general approach of peer to peer semantic wikis. It details a data model and editing operations. The next section defines the intention on this new data model.

In this work, the P2P semantic wiki embeds semantic data directly in the text as in SMW [34]. Therefore, they are replicated as a side effect of text replication as explained later.

A P2P semantic wiki network is composed of a set of interconnected autonomous semantic wiki servers (called also peers or nodes) that can dynamically join and leave the network. It is a truly decentralized unstructured P2P system which does not require central coordination or knowledge. It relies on a symmetric communication model where every peer may act as both a server and a client. Data management is based on optimistic data replication where every peer hosts a copy of the wiki pages and the associated semantic data. Every peer can autonomously offer all the services of a semantic wiki server, access, search and queries are executed locally without any queries routing on the network.

When a peer updates its local copy of data, it generates a corresponding operation. This operation is processed in four steps:

1. It is executed immediately against the local replica of the peer,
2. it is broadcasted through the P2P network to all other peers. Operations broadcasting is not in the scope of the paper, it can be realized by epidemic propagation[12] with anti-entropy protocol [10].
3. it is received by the other peers and
4. it is integrated to their local replica. If needed, the integration process merges this modification with concurrent ones, generated either locally or received from a remote server.

3.1 Data Model

The data model is an extension of WOOKI [35] data model to take in consideration semantic data. Every semantic wiki peer is assigned a global unique identifier named $NodeID$. These identifiers are totally ordered. As in any wiki system, the basic element is a wiki page and every wiki page is assigned a unique identifier $PageID$, which is the name of the page. The name is set at the creation of the page. If several servers create concurrently pages with the same name, their content will be directly merged by the synchronization algorithm. Notice that a URI can be used to unambiguously identify the concept described in the page. The URI must be global and location independent in order to ensure load balancing. For simplicity, in this paper, we use a string as page identifier.

3.1.1 Pages storage model

Definition A semantic wiki page *Page* is an ordered sequence of lines $L_B L_1, L_2, \ldots L_n L_E$ where L_B and L_E are special lines. L_B indicates the beginning of the page and L_E indicates the ending of the page.

Definition A semantic wiki line L is a four-tuple < LineID, content, degree, visibility > where

- $LineID$ is the line identifier, it is a pair of ($NodeID$, *logicalclock*) where $NodeID$ is the identifier of the wiki server and *logicalclock* is a logical clock of that server. Every server maintains a logical clock, this clock is incremented when an operation is generated. Lines identifiers are totally ordered so if $LineID_1$ and $LineID_2$ are two different lines with the values ($NodeID_1$, $LineID_1$) and ($NodeID_2$, $LineID_2$) then $LineID_1 < LineID_2$ if and only if (1) $NodeID_1 < NodeID_2$ or (2) $NodeID_1 = NodeID_2$ and $LineID_1 < LineID_2$.
- *content* is a string representing text and the semantic data embedded in the line.
- *degree* is an integer used by the synchronization algorithm, the degree of a line is fixed when the line is generated, it represents a kind of loose hierarchical relation between lines. Lines with a lower degree are more likely generated earlier than lines with a higher degree. By definition the degree of L_E and L_B is zero.
- *visibility* is a boolean representing if the line is visible or not. Lines are never really deleted they are just marked as invisible.

For instance, suppose there are two lines in a semantic wiki page about "France" , "France" is the identifier of the page.

France is located **in** [located In :: Europe]
The capital of France is [has Capital :: Paris]

The text is " France is located in [located In::Europe] The capital of France is [has Capital::Paris]." and the Semantic data are: "[located In::Europe]" and "[hasCapital::Paris]". As we can see, semantic data are considered as a text also. In this way, semantic data can be manipulated with text editing operations so there is no need to define specific editing operations for semantic data. Now, suppose these two lines are generated on the server with $NodeID = 1$ in the above order and there are no invisible lines, so the wiki page will be internally stored as:

L_B
((1,1), France is located **in** [located In :: Europe], 1, true)
((1,2), The capital of France is [has Capital :: Paris], 2, true)
L_E

Text and semantic data are stored in separate persistent storages. Text can be stored in files and semantic data can be stored in RDF repositories, as described in the next section.

3.1.2 Semantic data storage model

RDF is the standard data model for encoding semantic data. In P2P semantic wikis, every peer has a local RDF repository that contains a set of RDF statements extracted from its wikis pages. A statement is defined as a triple (Subject, Predicate, Object) where the subject is the name of the page and the predicates (or properties) and the objects are related to that concept.

For instance, the local RDF repository of the above server contains : $R = \{$(France, located In, Europe), (France, has Capital, Paris) $\}$. As for the page identifier, a global URI can be assigned to predicates and objects of a concept, for simplicity, we use a string.

We define two operations on the RDF repositories :

- insertRDF(R,t) : adds a statement t to the local RDF repository R.
- deleteRDF(R,t) : deletes a statement t from the local RDF repository R.

These operations are not manipulated directly by the end user, they are called implicitly by the editing operations as shown later.

3.2 Editing operations

There are two categories of editing operations, those defined to manipulate the wiki storage model and those used by the end user.

3.2.1 Storage model editing operations

There are two editing operations for editing the wiki text: *insert* and *delete*. An update is considered as a delete of old value followed by an insert of a new value.

There are no special operations for editing semantic data. Since semantic data are embedded in the text, they are edited with text operations. In this way, the RDF repositories are replicated and synchronized as a side effect of text replication and synchronization. Consequently, there is no need to define specific replication algorithms to replicate explicitly the RDF repositories.

1. *Insert(PageID, line, l_P , l_N)* where
   - $PageID$ is the identifier of the page of the inserted line,
   - $line =<$ LineID, content, degree, visibility $>$ is the line to be inserted.
   - l_P is the identifier of the line that precedes the inserted line and l_N is the identifier of the line that follows the inserted line.

   l_P and l_N represents the intention of the *insert* operation as defined in [35].

   During the insert operation, the semantic data embedded in the line are extracted, RDF statements are built with the page name as a subject and then they are added to the local RDF repository thanks to the function $insertRDF(R, t)$.

2. The *delete(PageID, LineID)* operation sets visibility of the line identified by $LineID$ of the page $PageID$ to false. The line is not deleted physically, it just marked as deleted. The identifiers of deleted lines must be kept as a tombstone. Tombstones, also known as "death certificate", are heavily used in optimistic replication, especially in Usenet [28]. Keeping these tombstones allow to keep trace of all changes. Semantic Media Wiki makes the same choice by keeping all the versions of the articles.

   During the delete operation, the set of the RDF statements contained in the deleted line are deleted from the local RDF repository thanks to the $deleteRDF(R, t)$.

For instance, if the peer with $NodeID = 1$ and a logical clock equals to 0 generates the operation: *op*= insert("France",< (1,1), " France is located in [located In::Europe]", 1, true >, L_B,L_E), during the insert operation, the semantic data [located In::Europe] are extracted and inserted to the local RDF Repository of peer 1. In other words, during the insertion operation the local RDF repository is augmented with the inserted semantic data. In the same way, if the peer 1 generates *op*= delete(" France ", $(1, 1)$), then the visibility of the line $(1, 1)$ is set to false and the statement $(France, locatedIn, Europe)$ is deleted from the local RDF repository.

3.2.2 User editing operations

A user of P2P semantic wiki does not edit directly the data model. Instead, she uses traditional wiki editing operations, when she opens a wiki page, she sees a view of the model. In this view, only visible lines are displayed. As in a traditional semantic wiki, she makes modifications i.e. adds new lines or deletes existing ones and she saves the page(s).

To detect user operations, a diff algorithm is used to compute the difference between the initial requested page and the saved one. Then these operations are transformed into model editing operations. A *delete* of the line number n is transformed into a *delete* of the n^{th} visible line and an *insert* at the position n is transformed into *insert* between the $(n-1)^{th}$ and the n^{th} visible lines. These operations are integrated locally and then broadcast to the other servers to be integrated.

Summing up, any user-triggered insert or delete operation leads to the following steps:

1. Transforming user operation into storage model operation.
2. Executing the transformed editing operation locally. During the integration of this operation, semantic data are extracted from the line (parsed). The semantic data are transformed into RDF triples. The local RDF repository is updated with the RDF triples.
3. Broadcasting the transformed editing operation through the P2P network,
4. Receiving the operation by the other peers and finally, integration to their local replica.

4 Correction Model

This section defines the intention of the editing operations for the P2P semantic wiki data model and shows how it is possible to preserve these intentions. A replicated cooperative editing system is said to be consistent if it always maintains the CCI properties[31].

4.1 Causality preservation:

The causality property ensures that operations ordered by a precedence relation will be executed in the same order on every server. In WOOT, the precedence relation relies on the *semantic causal dependency*. This dependency is explicitly declared as preconditions of the operations. Therefore, operations are executed on a state where they are legal *i.e.* preconditions are verified. In the following, we define causality for editing operations that manipulate text and RDF data model.

Definition insert Preconditions Let $Page$ be the page identified by $PageID$, let the operation *op=Insert(PageID, newline, p , n)*, *newline* =< LineID, c, d, v> generated at a server $NodeID$, R is its local RDF repository. The line *newline* can be inserted in the page $Page$ if its previous and next lines are already present in the data model of the page $Page$.

$$\exists i\ \exists j\ LineID(Page[i]) = p \qquad \wedge LineID(Page[j]) = n$$

Definition Preconditions of delete operation Let $Page$ be the page identified by $PageID$, let op = *Delete(PageID, dl)* generated at a server $NodeID$ with local RDF repository R, the line identified by *dl* can be deleted (marked as invisible), if its *dl* exists in the page.

$$\exists i\ LineID(Page[i]) = dl$$

For instance, there is a causal dependency between the operation op_1= *Insert("France",< (1,1), " France is located in [located In::Europe]", 1, true >,* L_B,L_e*)* and the operation op_2= *Insert("France",< (1,2), " The capital of France is [has Capital::Paris]", 1, true >, (1,1),*L_e*)*. This dependency can be deduced from the preconditions of the operation op_2. The precondition of op_2 requires the existence of the line identified by $(1, 1)$, therefore, the relation $op_1 \prec op_2$ (op_1 precedes op_2) must be respected at every server. When a server receives an operation, the operation is integrated immediately if its pre-conditions are evaluated to true else the operation is added to a waiting queue, it is integrated later when its pre-conditions become true.

4.2 Convergence

P2P semantic wikis ensure that when the same set of operations have been executed at all sites :

- all replicas of wikis pages are identical,
- all replicas of RDF repositories are identical.

4.3 Intentions and Intentions preservation

The intention of an operation is the visible effect observed when a change is generated at one peer, the intention preservation means that the intention of the operation will be observable on all peers, in spite of concurrent operations.

4.3.1 Operations Intentions

The intention of an operation depends on the type of manipulated data, it is a kind of a post-condition of the operation. The definition of the intention is not an easy task, according to this definition it is possible to preserve this intention or not. At first, we can define the intention of insert and delete operations in a simple way.

The intention of an insert operation *op= Insert(PageID, newline, p , n)* when generated at site $NodeID$, where *newline* =< nid, c, d, v> is defined as: (1) The content is inserted between the previous and the next lines and (2) the semantic data in the line content are added to the RDF repository of the server. The intention of a delete operation *op= delete(pid, l)* when generated at site S is defined as : (1) the line content of the operation is set to invisible and (2) the semantic data in the line content is deleted from the RDF repository of the server.

4.3.2 Intention preservation

Unfortunately, it is not possible to preserve the previous intention definitions. Assume that three P2P semantic wiki servers, $peer_1$, $peer_2$ and $peer_3$ share a semantic wiki page about "France". Every server has its copy of shared data and has its own persistence storages repositories. At the beginning, the local text and the RDF repositories are empty. At $peer_1$, $user_1$ inserts the line "France is located [located In::Europe]" at the position 1 in her copy of the "France" page. Concurrently, at $peer_2$ $user_2$ inserts a new line "France is a country in [located In::Europe]" in her local copy of "France" page at the same position and finally at $peer_3$ $user_3$ deletes the line added by $user_1$ as shown in the figure 3.

On $peer_1$ the operation op_1 is locally integrated, the wiki page contains the text line : "France is located [located In::Europe]", the semantic data "[locatedIn::Europe]" is stored in the local RDF repository as a triple. Then op_2 is received and integrated by the WOOT algorithm resulting an integration of the content of op_2 at line 2. op_1 is integrated before op_2, WOOT algorithm uses the order relation on the sites identifiers to serialize concurrent operations [20]. The "[located In::Europe]" property is now present twice in the wiki page and once in the local RDF repository. Finally, op_3 is received and integrated, the line inserted by the operation op_1 is deleted with its associated semantic data, therefore the RDF repository of $site_1$ is empty now. A similar result is obtained on $peer_2$. On $peer_3$, op_1 is received and integrated. $user_3$ decides to delete the line inserted by the op_1, op_3 is

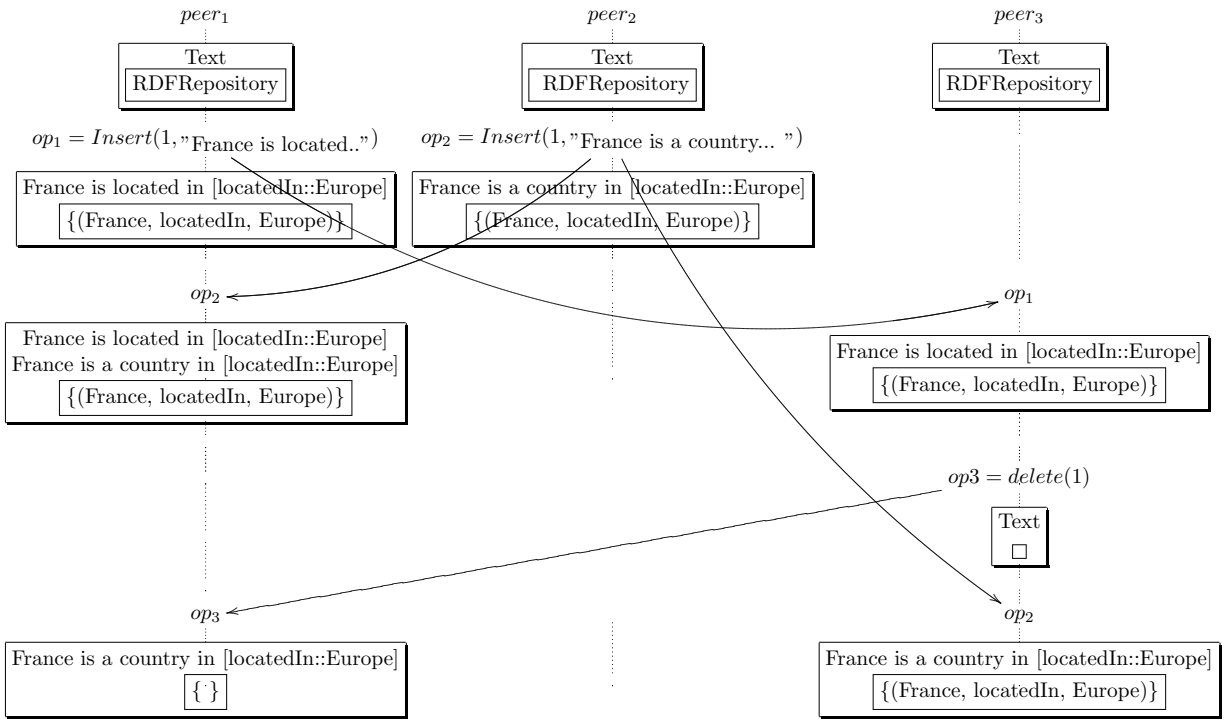

Figure 3: Semantic inconsistency after integrating concurrent modifications

locally integrated so " [located In::Europe]" property is deleted from the RDF repository of $site_3$ and then op_3 sent to other sites in order to be integrated. Finally, op_2 is received and integrated.

After the execution of the three operations on $peer_3$, the property "[located In::Europe]" is present in the wiki page and in the local RDF repository. However, this property is present in the wiki page of $peer_1$ and $peer_2$ but it is absent from the local RDF repository. As we can see, at the final state, the content of the wiki pages is the same but the RDF repositories of the peers diverge *i.e* they do not have the same set of RDF statements. This divergence results from the violation of the intentions of operations. The intention of the operation op_1 is to to add the "[locatedIn::Europe]" to the wiki page and to the RDF repository independently of concurrent operations, but this intention was violated by the concurrent delete operation.

The scenario above shows a kind of inconsistency between the semantic data in the wiki pages and the semantic data in the RDF repository. This is because RDF repositories are defined as a set of RDF statements, this introduces inconsistency among RDF repositories in a P2P semantic wiki.

4.4 Model for Intention preservation

To overcome inconsistency, we need to define some kind of "references integrity constraints" between semantic data in text and in RDF repository. One way to do that is to transform the RDF repository into a multi-set of statements in order to reflect the effect of the insertion of each triple.

Definition RDF repository is the storage container for RDF statements, each container is a *multi-set of RDF statements*. Each RDF repository is defined as a pair (T, m) where T is a set of RDF statements and m is the multiplicity function $m : T \rightarrow \mathcal{N}$ where $\mathcal{N} = 1, 2.....$

For instance, the multi-set R = { ("France", "Located In", "Europe),("France", "Located In", "Europe),("France", "hasCapital", "Paris") } can presented by R = { ("France", "Located In", "Europe)^2, ("France", "hasCapital", "Paris")^1 } where 2 is the number of occurrences of the first statement and 1 is this of the second one.

Definition Intention of insert operation Let S be a P2P semantic wiki server, R is its local RDF repository and $Page$ is a semantic wiki page. The intention of an insert operation *op= Insert(PageID, newline, p , n)* when generated at site S, where *newline* =< nid, c, d, v> and T is the set (or multi-set) of RDF statements in the inserted line, is defined as: (1) The content is inserted between the previous and the next lines and (2) the semantic data in the line content are added to R.

$$
\begin{array}{ll}
\exists i & \wedge\, \exists\, i_P < i \;\; LineID(Page'[i_P]) = p \\
& \wedge\, \exists\, i_N \leq i \;\; LineID(Page'[i_N]) = n \\
& \wedge Page'[i] = newline \\
& \wedge\, \forall j < i \;\; Page'[j] = Page[j] \\
& \wedge\, \forall j \geq i \;\; Page'[j] = Page[j-1] \\
& \wedge R' \leftarrow R \uplus T
\end{array}
$$

Where $Page'$ and R' are the new values of respectively the page and RDF repository after the application of the insert operation at the server S and $\uplus$ is the union operator of multi-sets. If a statement in T already exists in R so its multiplicity is incremented else it is added to R with multiplicity one.

Definition Intention of delete operation Let S be a P2P semantic wiki server, R is the local RDF repository and $Page$ is a semantic wiki page. The intention of a delete operation *op= delete(PageID, ld)* where T is the set (or multi-set) of RDF statements in the deleted line, is defined as (1) the line *ld* is set to invisible and (2) the number of occurrence of the semantic data embedded in *ld* is decreased by one, if the this occurrence is equal to zero which means this semantic data are no more referenced in the page then the semantic data are physically deleted from the R.

$$\exists i \wedge PageID(Page'[i]) = l$$
$$\wedge\ visibility(Page'[i]) \leftarrow false$$
$$\wedge\ R' \leftarrow R - T$$

Where $Page'$ and R' are the new values of respectively the page and RDF repository after the application of the delete operation at the server S and $-$ is the difference of multi-sets. If statement(s) in T exists already in R so its multiplicity is decremented and deleted from the repository if it is equal to zero.

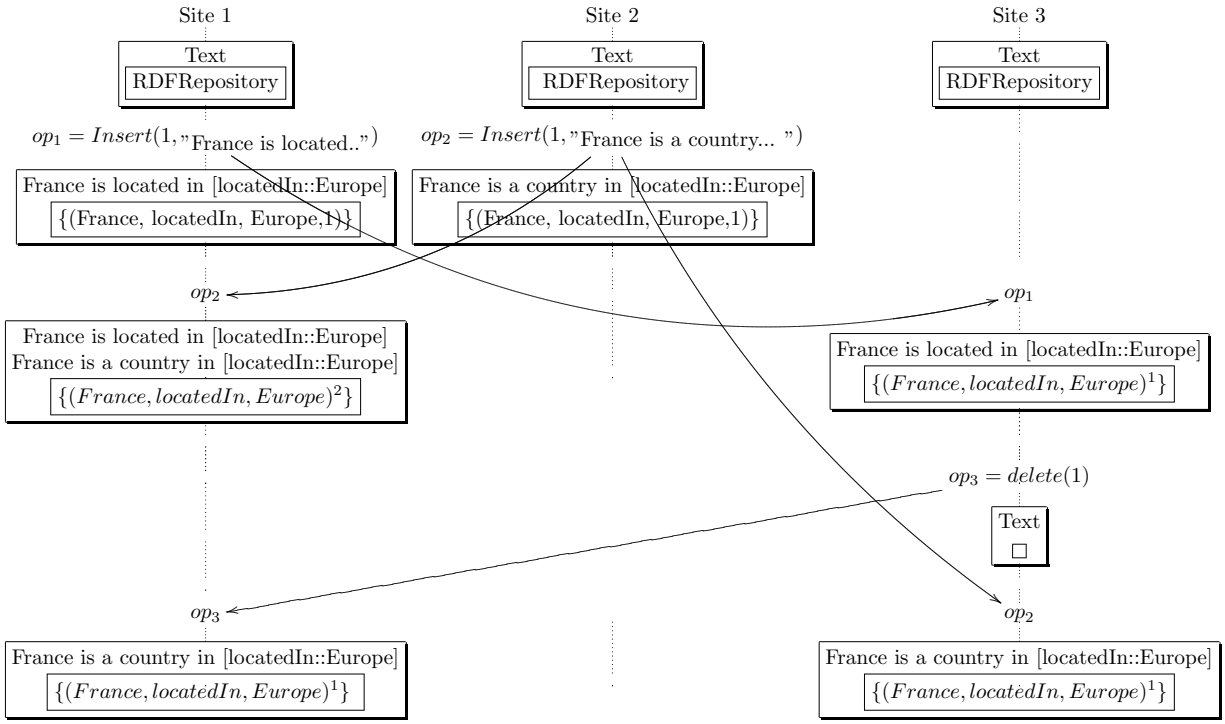

Figure 4: Convergence after integrating concurrent modifications

Let us consider again the scenario of the figure 3, as before on $site_1$, the operation op_1 is locally integrated, the local repository has the semantic data "[locatedIn::Europe]" with one occurrence. After reception and integration of op_2, the "[locatedIn::Europe]" property is now present twice in the wiki page so its multiplicity is 2. Finally, op_3 is received and integrated, the result is the deletion of the line inserted by the operation op_1 and the multiplicity of the property "[located In::Europe]" is decremented by one. A similar result is obtained on $site_2$.

After the integration of op_1 at $site_3$, the wiki page contains "France is located [locatedIn::Europe]", and the semantic data "[locatedIn::Europe]" is in the local repository with

one occurrence. The operation op_3 deletes the line 1 from the wiki page and decrements the number of occurrence of semantic data where occurrence becomes 0, so the triple is deleted from the RDF repository. Finally, op_2 is received and integrated and the result is the insertion of the content of op_2 and the "[locatedIn::Europe]" property is in RDF repository with one occurrence. By comparing the final result of the execution of the three operations on the three sites, we can see that the wiki pages and the RDF repositories converge by respecting the user intentions on text and semantic data. As we can see, the CCI properties are ensured for this new data type for all editing operations whereas most existing RDF synchronization algorithms handle the insertion of a triple in distributed RDF repositories [17, 7] without taking in consideration a possible deletion of a triple.

5 Algorithms

As any wiki server, a P2P semantic server defines *Save* operation which describes what happens when a wiki page is saved. In addition, it defines *Receive* and *Integrate* operations. The first describes what happens upon receiving a remote operation and the second integrates the operation locally.

5.1 Save operation

During saving a wiki page, a $Diff$ algorithm computes the difference between the saved and the previous version of the page and generates a patch. A *patch* is the set of delete and insert operations on the page ($Op = Insert(PageID,\ line,\ l_P\ ,\ l_N)$ or $Op = Delete(PageID,\ LineID)$). These operations are integrated locally and then broadcasted to other sites in order to be executed as shown below.

```
Upon Save(page, oldPage) :−
   let P ← Diff(page, oldPage)
 for each op ∈ P do
   Receive(op)
 endfor
Broadcast(P)
```

The $Broadcast(P)$ disseminates the patch as in[35]. It implements a lightweight probabilistic broadcast algorithm Lpbcast [12] with an anti-entropy algorithm [10]. The first disseminates quickly the updates over the P2P network and manages membership, the second recovers missing updates for sites that were off-line or crashed.

5.2 Delivery Operation

When an operation is received its preconditions are checked. If they are not satisfied, the operation is added to the waiting log of the server, else according to the type of the operations some steps are executed.

```
Upon Receive(op) :−
 if isExecutable(op) then
   if type(op) = insert then
     IntegrateIns(op)
   if type(op) = delete then
     IntegrateDel(op)
 else
      waitingLog ← waitingLog ⋃ {op}
 endif
```

The waiting log is visited after the integration and the operations that satisfy their preconditions are removed from the log and integrated.

```
isExecutable(op) :−
if type(op) = del then
   return containsL(PageID,LineID) and isVisible(LineID)
else
   return ContainsL(PageID,l_P)
        and ContainsL(PageID, I_N)
endif
```

The function $ContainsL(PageID, id)$ tests the existence of the line in the page, it returns true if this is the case. The function $isVisible(LineID)$ tests the visibility of the line.

5.3 Integrate operation

The integration of an operation is processed in two steps: (1) text integration and (2) RDF statements integration.

```
IntegrateDel(LineID) :−
   IntegrateDelT(LineID)
   IntegrateDelRDF(LineID)
```

To integrate a *delete* operation, the visibility flag of the line is set to false whatever its content.

```
IntegrateDelT(LineID) :−
   Page[LineID]. visibility    ← false
```

To integrate RDF statements, a counter is used to implement a multi-set RDF repository. A counter is attached to every RDF triple, the value of the counter corresponds to the number of occurrence of the triple in the repository.

```
IntegrateDelRDF(LineID) :−
let   S ← ExtractRDF(LineID)
if  S ≠ ∅ then
 for each  triple  ∈ S do
      triple .counter−−
    if    triple .counter == 0 then
      deleteRDF(R,triple)
    endif
endif
```

During the delete operation, the counter of the deleted statements are decreased, if the counter is zero the statements are physically deleted from the repository.

To integrate an insert operation $op = insert(\ PageID, line, l_P, l_N)$, the *line* has to be placed among all the lines between I_P and I_N, some of these lines can be previously deleted or inserted concurrently and the inserted semantic data are integrated.

```
 IntegrateIns(PageID, line,  l_P  ,  l_N) :−
    IntegratedInsT(PageID, line,  l_P  ,  l_N)
    IntegrateInsRDF(line)
```

To integrate a line in a wiki page, we use the integration algorithm defined in[35].

This algorithm selects the sub-sequence S' of lines between the previous and the next lines, in case of an empty result, the line is inserted before the next line. Else, the sub-sequence S' is filtered by keeping only lines with the minimum degree of S'. The remaining lines are sorted according to the line identifiers order relation $<_{id}$ [20], therefore, *line* will be integrated in its place according $<_{id}$ among remaining lines, the procedure is called recursively to place *line* among lines with higher degree in S'.

To integrate the semantic data, the RDF statements of the inserted line are extracted and added to the local RDF repository. If the statements exist already in the repository, their counter is incremented, otherwise, they are inserted into the RDF repository with a counter value equals to one as shown below.

```
IntegrateInsT(PageID, line, l_P , l_N) :−
let S'  ← subseq(Page[PageID]), l_P, l_N)
if S    = ∅ then
  insert(PageID, line, l_N)
else
   let i  ← 0
   let  d_min ← min(degree(S'))
   let F ← filter(S', degree = d_min)
   while (i < |F| − 1) and (F[i] <_id l) do
      i  ← i +1
    IntegrateInsT(PageID, line, F[i−1], F[i])
```

```
IntegrateInsRDF(line) :−
let  S ← ExtractRDF(line)
if S ≠ ∅ then
  for each triple ∈ S do
    if  Contains(triple) then
        triple .counter++
    else
       insertRDF(R,triple)
    endif
endif
```

5.4 Correctness

Theorem 1 *The integration algorithm preserves intention.*

The intention preservation for a text is demonstrated in [20]. Here, we are concerned with the intention of semantic data as defined in 4.3. The intention of an *insert* operation is trivially preserved by the algorithm *IntegrateInsRDF*. Since a possible way to implement a *multi-set* is to associate a counter to every element. In the same way, the algorithm *IntegrateDelRDF* preserves the intention of the *delete* operation.

6 Implementation and discussion

We have implemented the first peer to peer semantic wiki called SWOOKI based on algorithms presented in section 5. The SWOOKI prototype has been implemented in Java as servlets in a Tomcat Server. This prototype is available with a GPL license on sourceforge at `http://sourceforge.net/projects/wooki` and it is also available online at : `http://wooki.loria.fr/wooki1`.

6.1 SWOOKI Architecture

A SWOOKI server is composed of the following components (see figure 5):

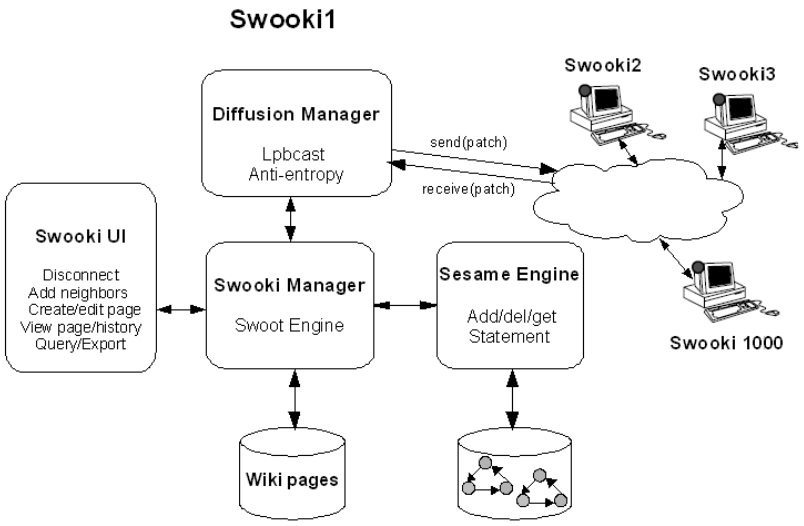

Figure 5: SWOOKI Architecture

User Interface. The SWOOKI UI component is basically a regular Wiki editor. Its only difference with a regular wiki is that it sends to the SWOOKI manager patches containing SWOOKI operations. So this component is able to generate patches of SWOOKI operations.

SWOOKI Manager. The SWOOKI manager implements the SWOOKI algorithm. Its main method is `Integrate(Patch)` that calls the `Receive()` algorithm for all operations contained in the patch.

Sesame Engine. The Sesame engine is a RDF store. It is controlled by the SWOOKI manager for storing and retrieving RDF triples. All requests are delegated to the Sesame engine. We use a facility of the Sesame interface to represent RDF triples as multi-set. Sesame engine is responsible of parsing, searching, storing and extracting semantic data. We use Sesame 2.0 [5] as RDF repository. Users can run SPARQL and SeRQL queries. This feature is provided by the query module in Sesame. This component allows also generating dynamic content for wiki pages using queries embedded in the wiki pages. It provides also a feature to export RDF graphs.

Diffusion Manager. In order to ensure the CCI model, we made the hypothesis that all operations eventually reach all sites of the unstructured P2P network. The diffusion manager is in charge to maintain the membership of the unstructured network and to implement a reliable broadcast. Membership and reliable broadcast of operations are ensured by an implementation of the LpbCast algorithm [12]. This algorithm ensures that all connected sites receive messages and that there is no partition in the P2P network. Of course, disconnected sites cannot be reached. So we added an anti-entropy mechanism based on [10]. The anti-entropy algorithm selects randomly a neighbor in the local table of neighbors and sends a digest of its own received messages. The receiver returns missing messages to caller. Using the anti-entropy implies that each server keeps received messages in a log, as this log can grow infinitely, the log is purged as detailed in [35].

6.2 Discussion

Every SWOOKI server provides all the services of a semantic wiki server. We analyze our system with respects to the following criteria, more detailed analyzing results about the implemented algorithms can be found in [12] and [13].

Availability, fault tolerance and load balancing Data are available on every peer so they are accessible even when some of the peers are unavailable. The global naming of the wiki pages (concepts) and their associated properties and objects and the respect of the CCI model ensure to have the same data at any node. So if a server is unavailable or slow, it is possible to access to another server.

Performance We analyze the performance with respects to messages necessary to execute query, propagate modification and synchronize data.

- Query execution : Every server can execute the query locally without generating network traffic for resolving them. Clearly, totally replicated P2P semantic wikis have better performance than partially replicated ones. In partially replicated P2P semantic wikis in general queries can not be executed locally, queries have to be routed to the peer that holds the data and the results have to be forwarded to the requested peer.
- Messages delivery: the complexity of the dissemination algorithm is computed as the number of rounds. A round is composed of three steps: (1) peer sends messages (patches) to its neighbors, (2) receives all messages sent to it and (3) carries out some local computation. The complexity of the dissemination algorithm as demonstrated in [12] is one round.
- Data synchronization: The integration algorithm does not need any message exchange to establish convergence and the convergence state is independent of the order of reception of messages, one round of communication when peers send and receive all messages is sufficient to obtain convergence.The complexity of the integration of n operations is $O(n^2)$ [13].

Scalability SWOOKI provides scalability with respects to the number of peers. Each peer knows only a fixed number of peers obtained randomly. It manages a table of neighbors that has a fixed size and contains a partial list of nodes in the P2P network which continuously evolves but never exceeds a fixed size. The dissemination algorithm updates this table during messages propagation where subscriptions and unsubscriptions attached to each message are used. It does not support solution for scalability with the size of data, the storage capacity is limited by the storage capacity of each node. We believe that this is not an obstacle for deploying a P2P semantic wiki since semantic wikis and wikis in general are community oriented, so a community can deploy a P2P semantic wiki where every member can has her own server or a group of members can share a P2P semantic wiki server. SWOOKI provides scalability with respect to the size of the community, a community with low funding can have a reliable knowledge sources and the possibility to work off-line and to make transactional changes.

Offline-work and transactional changes Users can work disconnected if they have no internet connection or if they decid to disconnect directly from the user interface. While disconnected, a user can change many semantic wiki pages in order to produce a consistent change. By this way she generates some sort of transactions. All changes performed in disconnected mode are kept in the diffusion manager component. As our optimistic replication algorithm forces all operations to commute (according to the CCI consistency) then, the concurrent execution of several transactions produce a consistent state in all case

Cost sharing The deployment of SWOOKI network is very similar to the deployment of the Usenet P2P network. A trusted peer of any organization can join the network, take a snapshot of replicated data and start answering wiki requests. The proposed architecture can be easily deployed on the internet across different organizations. In the contrast to the Wikipedia infrastructure that requires a central site with costly hardware and high bandwidth, our infrastructure can be deployed on any computer on the internet with a sufficient storage capacity. The cost of the underlying infrastructure can be shared by many different organizations.

7 Conclusion, Open Issues and Perspectives

Peer-to-peer semantic wikis combines both advantages of semantic wikis and P2P wikis. The fundamental problem is to develop an optimistic replication algorithm that ensures an adequate level of consistency, supports P2P constraints and manages semantic wiki page data type. We proposed such an algorithm in this paper. By combining P2P wikis and semantic wikis, we are able to deliver a new work mode for people working on ontologies: transactional changes. This work mode is particulary useful to help people to produce consistent changes in semantic wikis. Often, managing a semantic wiki requires to change a set of semantic wiki pages. These changes can take a long time and if intermediate state

results are visible, it can be confusing for other users and it can corrupt the result of semantic requests. We think that this working mode is crucial if we want to use semantic wikis for collaborative ontologies building.

However, this approach has many open issues and perspectives:

- Security issues are an important aspect. Replication makes security management more difficult. Often in wikis, security is represented as page attributes. If wiki pages are replicated, it means that security policies are replicated. In this case, it is possible to produce concurrent changes on security policy itself. If we re-centralize security management, we loose the benefits of replication. This problem can be solved by applying the approach proposed in this paper. To manage security policy : define the security policy data type, its operations, the intentions of these operations and update the replication algorithm with these new operations.
- In this paper, we mainly use replication for allowing transactional changes. As we maintain many replicas in this system, it is possible to distribute semantic queries computation among peers which reduces the load and allows the share of the computation cost.
- We explored also the combination of an unconstrained semantic wiki with a P2P wiki system based on total replication. We motivated this choice by pointing out the need of transactional changes. However, even if it is more difficult, it is possible to achieve the same objective with a P2P wiki based on partial replication and consequently take advantage of partial replication benefits such as reduced traffic, infinite storage and cheap join procedure.

References

[1] git based wiki: http://atonie.org/2008/02/git-wiki, 2008.

[2] World-wide web consortium: Resource description framework : http://www.w3.org/rdf, 2008.

[3] World-wide web consortium: Sparql query language for rdf:http://www.w3.org/tr/rdf-sparql-query, 2008.

[4] C. Bartelt. Consistence preserving model merge in collaborative development processes. In *CVSM '08: Proceedings of the 2008 international workshop on Comparison and versioning of software models*, pages 13–18, New York, NY, USA, 2008. ACM.

[5] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A generic architecture for storing and querying rdf and rdf schema. In *ISWC 2002: First International Semantic Web Conference*, 2002.

[6] M. Buffa, F. L. Gandon, G. Ereteo, P. Sander, and C. Faron. Sweetwiki: A semantic wiki. *Journal of Web Semantic*, 6(1):84–97, 2008.

[7] M. Cai and M. Frank. Rdfpeers: a scalable distributed rdf repository based on a structured peer-to-peer network. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 650–657, New York, NY, USA, 2004. ACM.

[8] M. Cart and J. Ferrie. Asynchronous reconciliation based on operational transformation for P2P collaborative environments. In *CollaborateCom: International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pages 127–138, White Plains, New York, USA, November 2008. IEEE Computer Society.

[9] P.-A. Chirita, S. Idreos, M. Koubarakis, and W. Nejdl. Publish/subscribe for rdf-based p2p networks. In *The Semantic Web: Research and Applications, First European Semantic Web Symposium, ESWS 2004*, pages 182–197. LNCS 3053, 2004.

[10] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic Algorithms for Replicated Database Maintenance. In *Proceedings of the ACM Symposium on Principles of Distributed Computing - PODC'87*, pages 1–12, Vancouver, British Columbia, Canada, August 1987. ACM Press.

[11] B. Du and E. A. Brewer. Dtwiki: a disconnection and intermittency tolerant wiki. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 945–952, New York, NY, USA, 2008. ACM.

[12] P. T. Eugster, R. Guerraoui, S. B. Handurukande, P. Kouznetsov, and A.-M. Kermarrec. Lightweight Probabilistic Broadcast. *ACM Transactions on Computer Systems*, 21(4):341–374, November 2003.

[13] C.-L. Ignat, G. Oster, P. Molli, M. Cart, J. Ferrié, A.-M. Kermarrec, P. Sutra, M. Shapiro, L. Benmouffok, J.-M. Busca, and R. Guerraoui. A Comparison of Optimistic Approaches to Collaborative Editing of Wiki Pages. In *Proceedings of the International Conference on Collaborative Computing: Networking, Applications and Worksharing - CollaborateCom 2007*, page 10, White Plains, New York, USA, November 2007. IEEE Computer Society.

[14] P. Johnson and R. Thomas. RFC677: The maintenance of duplicate databases, 1976.

[15] B. Kang, R. Wilensky, and J. Kubiatowicz. The hash history approach for reconciling mutual inconsistency. *Distributed Computing Systems, 2003. Proceedings. 23rd International Conference on*, pages 670–677, 2003.

[16] A.-M. Kermarrec, A. Rowstron, M. Shapiro, and P. Druschel. The IceCube Approach to the Reconciliation of Divergent Replicas. In *Proceedings of the ACM Symposium on Principles of Distributed Computing - PODC 2001*, pages 210–218, Newport, Rhode Island, USA, August 2001. ACM Press.

[17] C. Morbidoni, G. Tummarello, O. Erling, and R. Bachmann-Gmür. Rdfsync: efficient remote synchronization of rdf models. In *Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference (ISWC/ASWC2007), Busan, South Korea*, Berlin, Heidelberg, November 2007. Springer Verlag.

[18] J. Morris. DistriWiki: a distributed peer-to-peer wiki network. *Proceedings of the 2007 international symposium on Wikis*, pages 69–74, 2007.

[19] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmér, and T. Risch. Edutella: a p2p networking infrastructure based on rdf. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 604–615, New York, NY, USA, 2002. ACM.

[20] G. Oster, P. Urso, P. Molli, and A. Imine. Data Consistency for P2P Collaborative Editing. In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work - CSCW 2006*, Banff, Alberta, Canada, November 2006. ACM Press.

[21] C. L. Patrick Mukherjee and A. Schurr. Piki - a peer-to-peer based wiki engine. In *P2P08: Eighth International Conference on Peer-to-Peer Computing*, pages 185–186. IEEE, 2008.

[22] K. Petersen, M. J. Spreitzer, D. B. Terry, M. M. Theimer, and A. J. Demers. Flexible update propagation for weakly consistent replication. In *Proceedings of the sixteenth ACM symposium on Operating systems principles - SOSP'97*, pages 288–301. ACM Press, 1997.

[23] G. rald Oster, P. Urso, P. Molli, and A. Imine. Data consistency for p2p collaborative editing. In *Proceedings of the 2006 ACM Conference on Computer Supported Cooperative Work, CSCW 2006, Banff, Alberta, Canada, November 4-8, 2006*. ACM, 2006.

[24] Y. Saito and M. Shapiro. Optimistic Replication. *ACM Computing Surveys*, 37(1):42–81, 2005.

[25] S. Schaffert. Ikewiki: A semantic wiki for collaborative knowledge management. In *WETICE*, pages 388–396. IEEE Computer Society, 2006.

[26] M. Shapiro, K. Bhargavan, and N. Krishna. A constraint-based formalism for consistency in replicated systems. In *opodis*, number 3544, pages 331–345, Grenoble, France, December 2004.

[27] H. Spencer and D. Lawrence. *Managing Usenet*. January 1988.

[28] H. Spencer and D. Lawrence. *Managing Usenet*. O'Reilly Sebastopol, 1998.

[29] S. Staab and H. Stuckenschmidt, editors. *Semantic Web And Peer-to-peer*. Springer, 2005.

[30] G. Stahl, editor. *Group cognition: Computer support for building collaborative knowledge*. Cambridge, MA: MIT Press, 2006.

[31] C. Sun, X. Jia, Y. Zhang, Y. Yang, and D. Chen. Achieving Convergence, Causality Preservation, and Intention Preservation in Real-Time Cooperative Editing Systems. *ACM Transactions on Computer-Human Interaction*, 5(1):63–108, March 1998.

[32] D. B. Terry, M. M. Theimer, K. Petersen, and A. J. Demers. Managing update conflicts in bayou, a weakly connected replicated storage system. In *In Proceedings of the Fifteenth ACM Symposium on Operating Systems Principles*, pages 172–183, 1995.

[33] G. Tummarello, C. Morbidoni, J. Petersson, P. Puliti, and F. Piazza. Rdfgrowth, a p2p annotation exchange algorithm for scalable semantic web applications. In *P2PKM*, CEUR Workshop Proceedings, 2004.

[34] M. Vôlkel, M. Krtôzsch, D. Vrandecic, H. Haller, and R. Studer. Semantic wikipedia. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 585–594, New York, NY, USA, 2006. ACM Press.

[35] S. Weiss, P. Urso, and P. Molli. Wooki: a p2p wiki-based collaborative writing tool. In *Web Information Systems Engineering*, Nancy, France, December 2007. Springer.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399